# Data Communication and Net-Centric Computing

## COSC 1111/2061/1110

## Lecture 4

## Error Detection

# Lecture Overview

- During this lecture, we will understand
  - Error detection
  - Single bit error, burst error
  - Parity check
  - Two dimensional parity checks
  - CRC

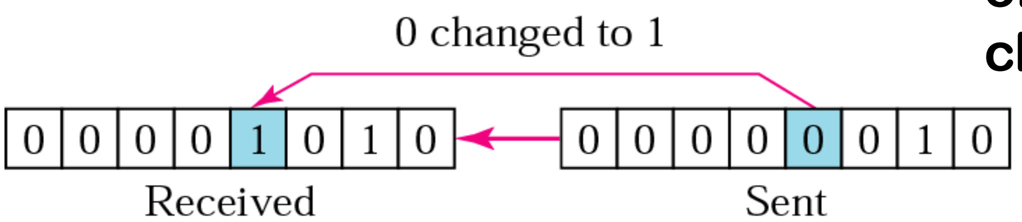
- Recommended reading
  - Chapter 6 (Stallings)

# Error Detection

- Transmission channel has impairments
  - Finite probability of an error
- Reliable communication
  - The receiver should be able to detect the presence of errors in received data
  - Error detected – a mechanism to recover correct information needed
- Backward error detection
  - Each transmitted byte contains additional information allowing receiver to check
  - Error detected – requests for re-transmission of data

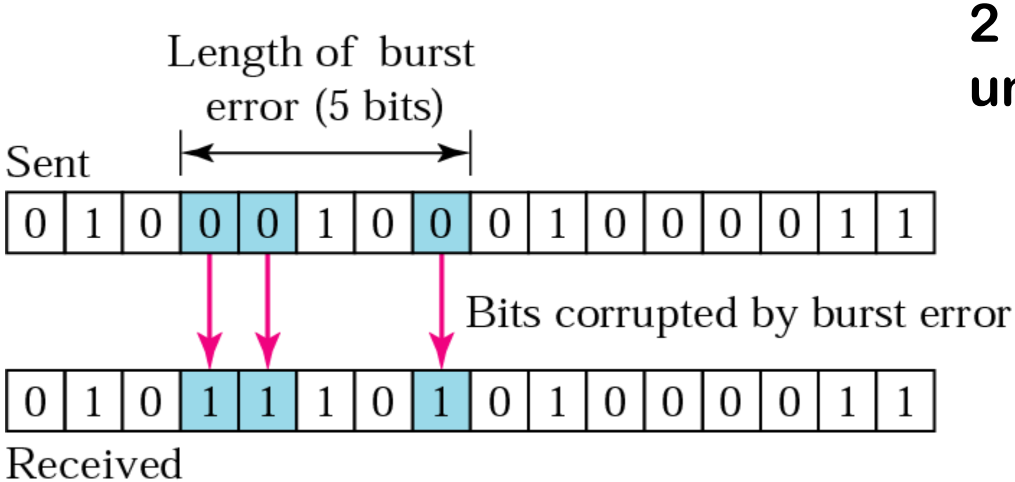
# Types of Errors

## Single-bit error




- In a single-bit error, only one bit in the data unit has changed.

## Burst errors




- A burst error means that 2 or more bits in the data unit have changed.

# Single Parity Check

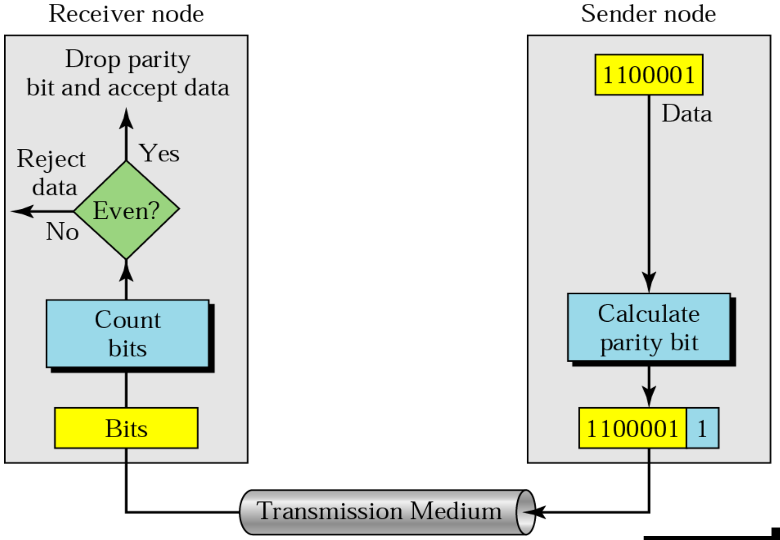
- Append an overall parity check to *k* information bits

Info Bits: $b_1, b_2, b_3, \ldots, b_k$

Check Bit: $b_{k+1} = b_1 + b_2 + b_3 + \ldots + b_k$ modulo 2

Codeword: $(b_1, b_2, b_3, \ldots, b_{k,}, b_{k+!})$

- All codewords have even # of 1s
- Receiver checks to see if # of 1s is even (or odd for odd-parity)
  - All error patterns that change an odd # of bits are detectable
  - All even-numbered patterns are undetectable
- Parity bit used in ASCII code



# Example of Single Parity Code

- Information (7 bits): (0, 1, 0, 1, 1, 0, 0)
- Parity Bit: $b_8 = 0 + 1 + 0 + 1 + 1 + 0 = 1$
- Codeword (8 bits): (0, 1, 0, 1, 1, 0, 0, 1)

- If single error in bit 3 : (0, 1, 1, 1, 1, 0, 0, 1)
  - # of 1's =5, odd
  - Error detected

- If errors in bits 3 and 5: (0, 1, 1, 1, 0, 0, 0, 1)
  - # of 1's =4, even
  - Error not detected



RMIT UNIVERSITY

# Two-Dimensional Parity Check

- More parity bits to improve coverage
- Arrange information as columns
- Add single parity bit to each column
- Add a final "parity" column
- Used in early error control systems

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |

Last column consists of check bits for each row

Bottom row consists of check bit for each column

# Error-detecting capability

One error

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |

Two errors

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |

Three errors

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |

Four errors (undetectable)

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |

- If 1, 2, or 3 errors occur anywhere in the matrix during transmission, then at least one row or parity check will fail.
- Not all patterns >4 errors can be detected

Arrows indicate failed check bits

# Other Error Detection Codes

- Many applications require very low error rate
- Need codes that detect the vast majority of errors
- Single parity check codes do not detect enough errors
- Two-dimensional codes require too many check bits
- The following error detecting codes used in practice:
  - Internet Check Sums
  - CRC Polynomial Codes

# CRC Arithmetic

- Polynomial arithmetic of modulo two
- Addition and subtraction in CRC are identical
- Nothing more than two binary numbers being XORed

$$\begin{array}{r} 01101100 \\ \underline{01001101} \\ 00100001 \end{array}$$

Addition:

$$(x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + x^6 + x^6 + x^5 + 1$$

$$= x^7 + (1+1)x^6 + x^5 + 1$$

$$= x^7 + x^5 + 1 \quad \text{since } 1+1=0 \text{ mod2}$$

Multiplication:

$$(x + 1)(x^2 + x + 1) = x(x^2 + x + 1) + 1(x^2 + x + 1)$$

$$= x^3 + x^2 + x) + (x^2 + x + 1)$$

$$= x^3 + 1$$

# Binary Polynomial Division

- **Division with Decimal Numbers**

$$
\begin{array}{r}
34 \\
35\ \overline{)\,1222} \\
105 \\
\hline
172 \\
140 \\
\hline
32
\end{array}
$$

34 ← quotient

1222 ← dividend

35 ← divisor

32 ← remainder

dividend = quotient x divisor +remainder

1222 = 34 x 35 + 32

- **Polynomial Division**

$$
\begin{array}{r}
x^3 + x^2 + x \qquad = q(x) \text{ quotient} \\
x^3 + x + 1\ \overline{)\, x^6 + x^5} \\
x^6 + \quad x^4 + x^3 \\
\hline
x^5 + x^4 + x^3 \\
x^5 + \quad x^3 + x^2 \\
\hline
x^4 + \quad x^2 \\
x^4 + \quad x^2 + x \\
\hline
x \qquad = r(x) \text{ remainder}
\end{array}
$$

$x^3 + x + 1$ ← divisor

$x^6 + x^5$ ← dividend

*Note: Degree of r(x) is less than degree of divisor*

# Cyclic Redundancy Check

- Common and most powerful error detecting code is CRC
  - Given a block of *k* bits, the transmitter generates *n* bit sequence called Frame Check Sequence (FCS)
  - Resulting frame *k+n* bits which is exactly divisible by some predetermined number
- Receiver divides frame by that number
  - If no remainder, assume no error
- Because of features of modulo-2 arithmetic, both transmitter and receiver use same divisor

# CRC Example – base 2 method

- Example
- Given data
  - 1101 1100 1110
- Pattern
  - 1101
- Frame Check Sequence
  - Pattern – 1 bits
  - 4 – 1 = 3 bits

**Sender side**

```
1101 | 1101 1100 1110  000
     | 1101
  ------------------------
            1100
            1101
  ------------------------
               1 111
               1 101
       -------------------
                  100 0
                  110 1
         -----------------
                   10 10
                   11 01
          ----------------
                     1 110
                     1 101
           ---------------
                       011
```

# CRC Example – base 2 receiver side

- Append the remainder to the given data
- Use the same pattern for division.
- If no remainder then transmission is error free

**receiver side**

```
1101 |1101 1100 1110 011
      1101
      ------------------
           1100
           1101
           -------------
               1 111
               1 101
               ---------
                  100 0
                  110 1
                  ------
                    10 11
                    11 01
                    -----
                     1 101
                      1101
                     -----
                      0000
```

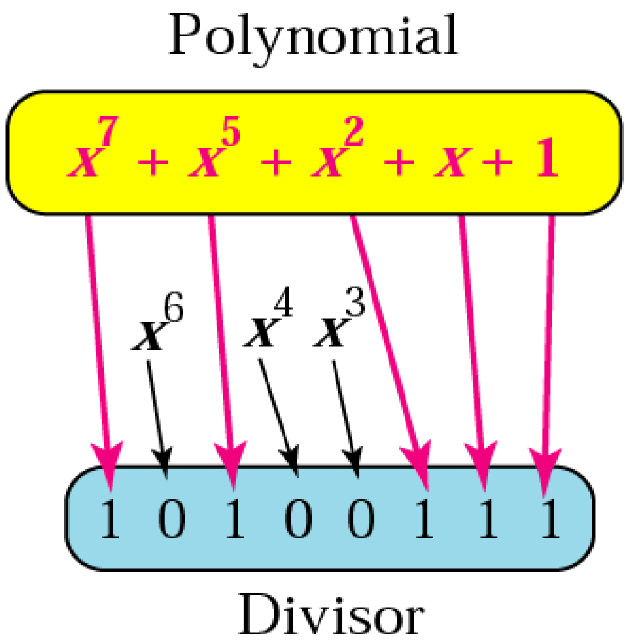




RMIT UNIVERSITY

# CRC – shifted poly method

- Essesntially same like base 2 method.
- Convert your given data and pattern to polynomial format. Example given below:




- Apply polynomial division and follow procedures that you learned in base 2 method

# Standard polynomials

| Name | Polynomial | Application |
|---|---|---|
| CRC-8 | $x^8 + x^2 + x + 1$ | ATM header |
| CRC-10 | $x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ | ATM AAL |
| ITU-16 | $x^{16} + x^{12} + x^5 + 1$ | HDLC |
| ITU-32 | $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ | LANs |

$$x^{16} + x^{15} + x^2 + 1$$

- CRC-16 detects
  - detects more errors than checksum
  - single and double errors
  - errors with an odd number of bits
  - burst errors of length 16 or less, 99.997% of 17-bit error bursts, and 99.998% of 18-bit and longer bursts.

# Checksum Method

**The sender follows these steps:**

- The unit is divided into k sections, each of n bits.
- All sections are added using one's complement to get the sum.
- The sum is complemented and becomes the checksum.
- The checksum is sent with the data.

**The receiver follows these steps:**

- The unit is divided into k sections, each of n bits.
- All sections are added using one's complement to get the sum.
- The sum is complemented.
- If the result is zero, the data are accepted: otherwise, rejected.

# Checksum Example –sender side

- Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001   00111001

- Using one's complement

| | |
|---|---|
| | 10101001 |
| | 00111001 |
| Sum | 11100010 |
| Checksum | 00011101 |

The pattern sent is   10101001   00111001   00011101

# Checksum Example – receiver side

- Now suppose the receiver receives the pattern sent and there is no error.

10101001   00111001   00011101

- When the receiver adds the three sections, it will get all 1s, which, after complementing, is all 0s and shows that there is no error.

| | |
|---|---|
| | 10101001 |
| | 00111001 |
| | 00011101 |
| Sum | 11111111 |
| Complement | 00000000 means that the pattern is OK. |

# Summary

- In this lecture, we have:
  - Understood Error detection
  - Parity Check, Checksum, CRC
  - Examples error detection methods

# Next Time

- We will know
  - more on Multiplexing

- Suggested Reading:
  - Chapter 8 (Stallings)